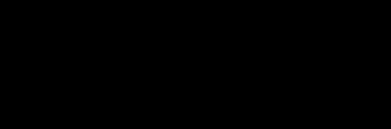


Audio and MIDI

Session 507

Audio and MIDI

Craig Keithley
USB and FireWire Technology Evangelist

Core Audio HAL

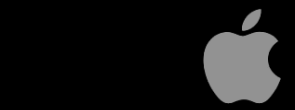
Jeff Moore
Core Audio Engineering

Introduction

- The Audio Hardware Abstraction Layer (HAL) provides the low-level access to audio devices attached to the system
- The HAL supports all kinds of devices from simple consumer devices to complex pro gear
    - Multi-channel
    - High sample rates
    - Non-linear PCM formats

Introduction to the HAL API

- Loosely object-oriented
- Each object has a set of properties that describe and manipulate its state and behavior



HAL Object Property Basics

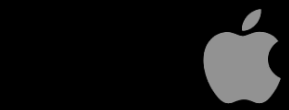
- Addressed as key/value pairs
  - The key varies from object to object, but always includes an integer ID
  - The value is an untyped block of memory whose contents are determined by the ID
  - ***GetPropertyInfo routines provide the size of a property and whether or not it can be set

HAL Object Property Basics

- Provides notifications for value changes
  - The client provides a ***PropertyListenerProc** to call when the value of a property changes
  - Wild cards can be used when adding Listeners
  - Listeners are called from the HAL's **CFRunLoop**
  - Changing a property is an asynchronous operation

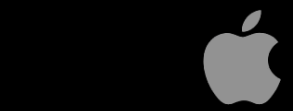
HAL System Object

- **AudioHardware*** routines
- Manages process global state
- Properties addressed by ID
- Important properties:
  - **kAudioHardwarePropertyDevices**
  - **kAudioHardwarePropertyRunLoop**
  - **kAudioHardwarePropertyUnloadingIsAllowed**



HAL Device Objects

- **AudioDevice***** routines
- The basic unit for I/O and timing
- Properties addressed by ID, direction, and channel
- Important properties:
  - **kAudioDevicePropertyBufferFrameSize**
  - **kAudioDevicePropertyStreamConfiguration**
  - **kAudioDevicePropertyStreamFormat**

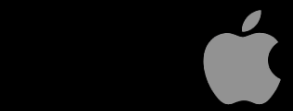


HAL Stream Objects

- **AudioStream*** routines
- Manage the format of a single I/O buffer of a device
- Properties addressed by ID and channel
- Important properties:
  - **kAudioStreamPropertyStartingChannel**
  - **kAudioDevicePropertyStreamConfiguration**
  - **kAudioStreamPropertyPhysicalFormat**

Example: Playing an AC-3 File

- AC-3 bitstreams are organized into packets
  - Each packet has 1536 frames
  - Each packet can vary in byte size
  - Each packet starts with a 16 bit sync word
  - AC-3 suitable for transport via S/PDIF has an additional 8 byte header



Demo

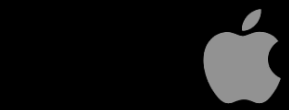
AC-3 Playback

Core MIDI

Doug Wyatt
Core Audio Engineering

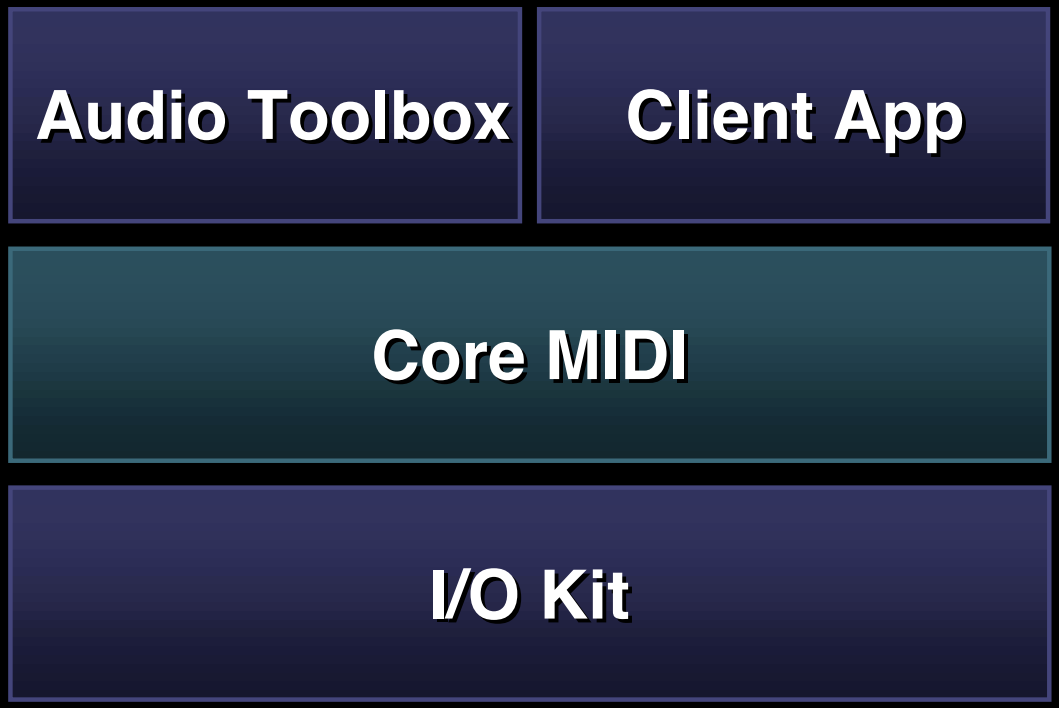
Agenda

- Introduction to Core MIDI
- Go through each concept in theCore MIDI API
- Cover areas of 10.1 API's that are sources of frequent questions
- Introduce new Jaguar features

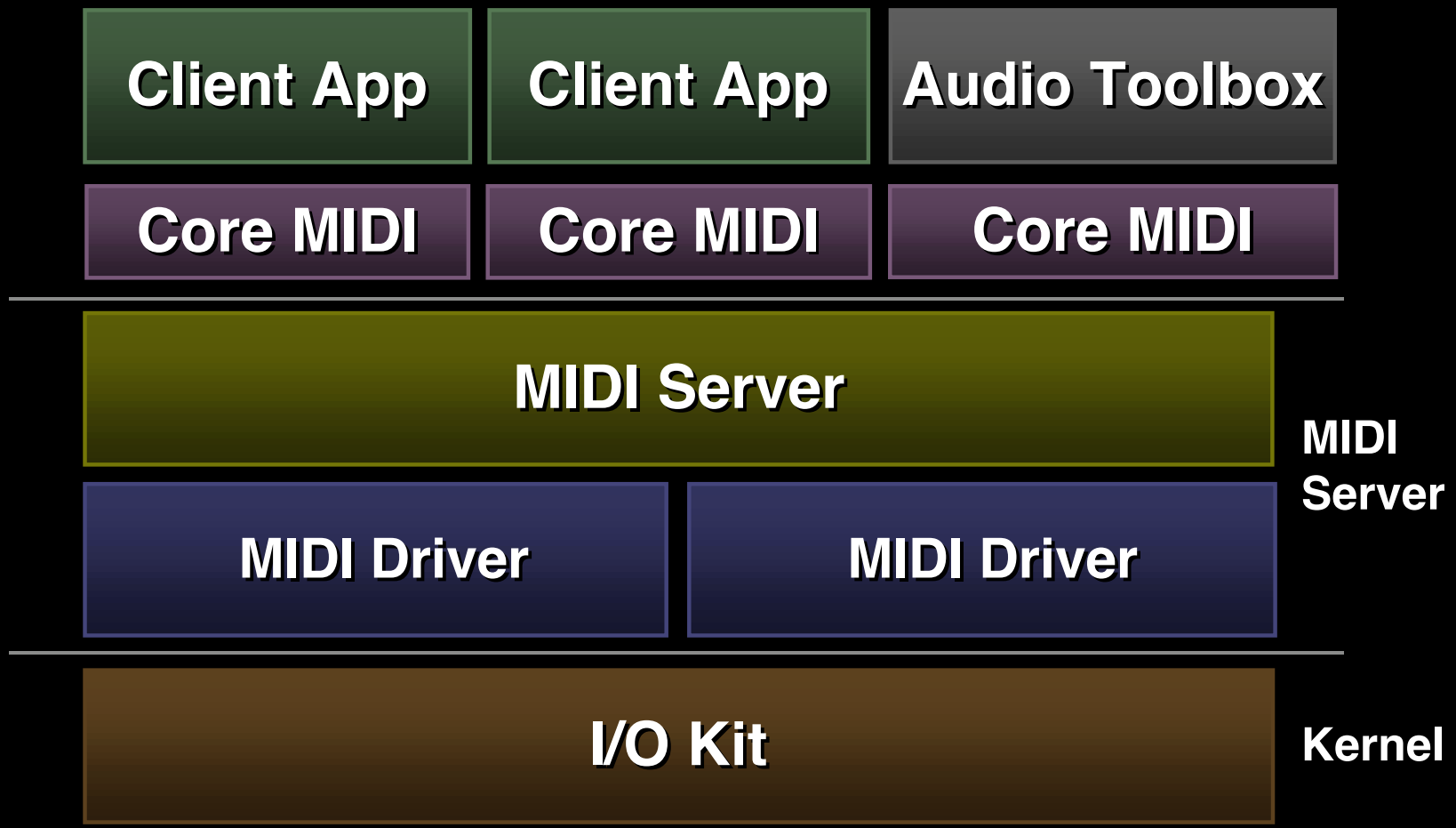


Core MIDI in the Big Picture

- MIDI services provide high-performance access to MIDI hardware devices



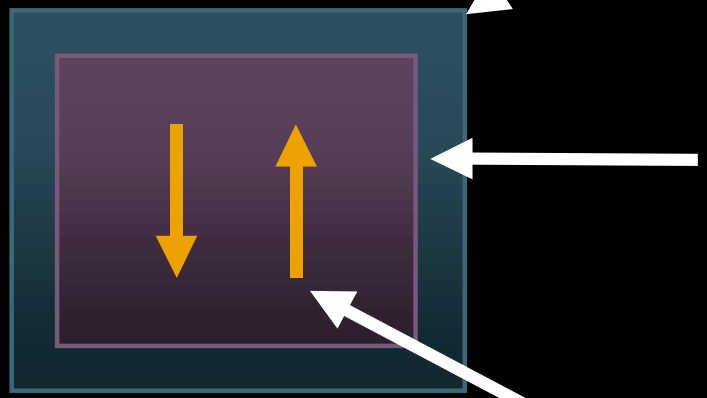
Core MIDI Implementation



Devices, Entities, Endpoints

- **MIDIDevice**, **MIDIEntity**, **MIDIEndpoint** objects
- Created by drivers
- Devices contain entities (logically distinct subcomponents), which contain endpoints (MIDI sources and destinations)

Device/Entity/Endpoint Properties



Device
Name = X999
Manufacturer = Xcorp
Model = X999

Entity
Name = Port1
Manufacturer = Xcorp
Model = X999
SysExID = 17

Endpoint
Name = Port1 In
Manufacturer = Xcorp
Model = X999
SysExID = 17

Property Use

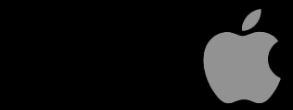
- Drivers set certain properties on their devices
- Apps should not touch these
- Apps may attach their own private properties

Displaying Endpoint Names

- First, ask for the endpoint name
- If it is unique, use that
- If not, prepend the device name

Two Ways to Find Endpoints

- Iterate through the source and destination endpoints
    - You must do this to see virtual endpoints
- Hierarchical walk (devices —> entities —> endpoints)
    - Use this for studio setup operations
    - Caution: offline and private objects, missing virtual endpoints



External Devices, Entities and Endpoints

- Are also **MIDIDevices**, **MIDIEntities**, **MIDIEndpoints** with properties
- Added to the system by the user
  - Audio MIDI Setup
  - Third-party studio setup editor
- Optional; system still works without them

Driver—Owned and External Objects

- Linked via **kMIDIPropertyConnectionUniqueID**
  - A single 32-bit unique ID or (as of Jaguar) an array of them
  - Set on a driver's endpoint to indicate a connection to an external endpoint

Properties and External Objects

- External object's properties override those of the driver-owned object
- But you must perform that override yourself
    - Which name to use when a single MIDI port is connected to multiple external devices?

Informational Properties

- Many are new for Jaguar
- Transmit and receive MIDI channels
- Transmit and receive event types (channel messages, MTC, clock)
- Supports General MIDI, MIDI Machine Control
- See **MIDIServices.h** for complete list

More Property Changes

- Schedule-ahead support for virtual destinations (E.g., softsynths)
- Reference to icon file
- Creator can set the uniqueID of a virtual endpoint
- XML description of device's patch, note and control names

New Informational APIs

- Make navigating the setup easier

MIDIObjectFindByUniqueID

MIDIEntityGetDevice and **MIDIEndpointGetEntity**

Saving Object References

- Save the object's name and uniqueID
- When reading, attempt to resolve the uniqueID (**MIDIObjectFindByUniqueID**)
- If that fails, you can still display the old object's name while offering the user an alternate object

MIDIClient Objects

- Receives notifications of system state changes
- OK to have more than one per process
- Jaguar: finer-grained notifications
    - Object added
    - Object removed
    - Object property changed
    - If you handle these, ignore the older “world-changed” notification

MIDIPort Objects

- Communication channel to the server
  - Not a physical MIDI connector
- Multiple ports only needed in modular apps
  - Multiple output ports are merged in MIDI Server

MIDIThruConnection

- New for Jaguar
- Allows creating MIDI source-to-destination Thru connections in the server
    - Reduces IPC messaging to the client
- Fairly extensive MIDI filtering/mapping

Patch, Note and Control Names

- Defined in a **MIDINameDocument** XML file
  - Still in Preliminary Draft form
- Allows users to choose sounds by name

```
<patchNameList name="Programs 00">
        <patch number="P0-0" name="None">
            <MIDI>
                <ProgramChange progNum="0" />
            </MIDI>
        </patch>
        <patch number="P0-1" name="3rd World Order" />
        <patch number="P0-2" name="Stereo Grand" />
    </patchNameList>
```

Using MIDINameDocuments

- Not trivial; Core MIDI just provides document references, you parse the XML
- Still in the process of finalizing the format
    - No current documents available for public review

Demo

Audio MIDI Setup

Multithreading Issues

- **CoreMIDI** (used by apps) is completely thread-safe
  - May call any function from any thread
- **CoreMIDIServer** (used by drivers) is not
  - Drivers must make all calls except I/O on main thread

Multithreading Issues

- All client **MIDIInputPorts** and their **ReadProcs** are serviced by a single thread
  - Time-constraint policy on 10.1
  - Fixed priority 63 on Jaguar
- All notification callbacks called from the run-loop (thread) on which **CoreMIDI** was first called

Audio, MIDI and Thread Priorities

William Stewart
Core Audio Engineering

Introduction

- Thread behaviour
  - Priorities
  - Policies
- Hardware I/O

Primary Interrupts

- Used by the audio drivers to:
  - Supply accurate timing information to the HAL
  - The HAL uses that information to schedule an application's I/O threads
- Audio Drivers:
  - Use the I/O threads of the clients to package their data (mix/clip)

Time Constraint Threads

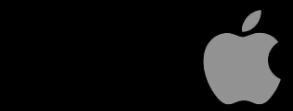
- The I/O threads of the AudioDevice
  CoreAudio/AudioHardware.h
- The I/O threads of the MIDI Server
- Highest priority threads on the system
  - Run at a fixed priority of 96

Kernel Tasks

- Kernel uses a pool of threads to run its tasks
  - Run at a fixed priority of 80

Mac OS Services/ Application Threads

- Fixed Priority threads
  - Jaguar
    - "Special privileges"
      - Preempt kernel tasks
      - Makes these a pseudo "real-time" threads
  - Highest priority is 63-Fixed Priority
- Time Share threads
  - Priority degrades the longer a thread runs



Mac OS Services/ Application Threads

- Window Server (Jaguar)
  - Has (at least) 2 fixed priority threads
    - 63–very little work—fields events
      - Will dispatch any substantial work to:
    - 51–fields requests from applications (E.g., FindWindow...) and does compositing
      - One of these per processor
  - Can involve transitions to the kernel
    - Thus—need to preempt those activities

Mac OS Services/ Application Threads

- Fixed Priority threads
  - Carbon
    - 54–Time Manager
    - 53–Asynch File I/O
    - 52–Deferred Task
    - MP Threads are currently Time Share threads
      - Will have an API in Jaguar to set their policy and priority

How It All Lines Up

- System/Kernel
  - Primary Interrupt
  - Scheduler
  - T/C threads (p=96)
    - Audio I/O (IOProc)
    - MIDI I/O (MIDI Server)
  - Kernel tasks (p=80)
  - MIDI Client thread (fp=63)

How It All Lines Up

- Application
  - Event thread of Window Server (fp=63)
  - Carbon Time Manager (fp=54)
  - Carbon Asynch File I/O (fp=53)
  - Carbon Deferred Task (fp=52)
  - Carbon MP threads (user-definable)
  - Window Server user event processing and compositing (fp=51)
  - Main thread of app (time share=36)

Demo

“Million Monkeys”

Conclusion

- Look for new APIs in the Jaguar frameworks
- Some apps using both Core Audio and MIDI features are shipping
- Thanks for your feedback!
    - You have been driving the new features
    - Please keep it up

Roadmap

502 Core Audio Technologies	Room J **Tue., 2:00pm**
508 Audio Units and Audio Converter Components: Writing Audio Units and Audio Codecs	Room J **Wed., 5:00pm**
607 QuickTime and MPEG-4: Short Overview of AAC	Room A2 **Fri., 3:30pm**
FF025 Audio and MIDI: Question and Answer Forum	Room J **Fri., 3:30pm**

Who to Contact

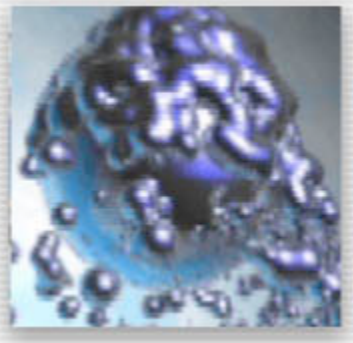
Craig Keithley
USB and FireWire Technology Evangelist
Keithley@apple.com

http://developer.apple.com/wwdc2002/urls.html

For More Information

- Core Audio Developer Services
- Mailing List—Core Audio API
  - **http://lists.apple.com/**
- SDKs
  - **http://developer.apple.com/audio/**

Q&A



Craig Keithley
USB and FireWire Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002